

What Is Claimed Is:

1 1. A method to facilitate code verification and garbage collection in a
2 platform-independent virtual machine, comprising:
3 receiving a code module written in a platform-independent language;
4 examining the code module to locate a call to a program method within the
5 code module; and
6 transforming the code module so that all operands remaining on an
7 evaluation stack when the program method is called relate to the program method;
8 whereby verification and garbage collection of the code module is
9 simplified.

1 2. The method of claim 1, wherein transforming the code module
2 involves ensuring that local variables hold only values of a single type and do not
3 hold variables of different types at different times.

1 3. The method of claim 1, wherein transforming the code module
2 involves ensuring that the evaluation stack includes only elements related to a
3 bytecode that may trigger garbage collection when the bytecode is executed.

1 4. The method of claim 1, wherein transforming the code module
2 involves ensuring that only parameters for the program method are on the
3 evaluation stack when the program method is called.

1 5. The method of claim 1, wherein transforming the code module
2 further comprises spilling to memory stack slots that do not include operands for
3 the call to the program method.

1 6. The method of claim 5, further comprising filling stack slots that
2 were previously spilled upon return from the program method.

1 7. The method of claim 6, wherein the program method is associated
2 with a single typemap to indicate a type for each variable on the evaluation stack.

1 8. An apparatus to facilitate code verification and garbage collection
2 in a platform-independent virtual machine, comprising:
3 a receiving mechanism configured to receive a code module written in a
4 platform-independent language;
5 an examining mechanism configured to examine the code module to locate
6 a call to a program method within the code module; and
7 a transforming mechanism configured to transform the code module so
8 that all operands remaining on an evaluation stack when the program method is
9 called relate to the program method;
10 whereby verification and garbage collection of the code module is
11 simplified.

1 9. The apparatus of claim 8, wherein transforming the code module
2 involves ensuring that local variables hold only values of a single type and do not
3 hold variables of different types at different times.

1 10. The apparatus of claim 8, wherein transforming the code module
2 involves ensuring that the evaluation stack includes only elements related to a
3 bytecode that may trigger garbage collection when the bytecode is executed.

1 11. The apparatus of claim 8, wherein transforming the code module
2 involves ensuring that only parameters for the program method are on the
3 evaluation stack when the program method is called.

1 12. The apparatus of claim 8, further comprising a spilling mechanism
2 configured to spill to memory stack slots that do not include operands for the call
3 to the program method when transforming the code module.

1 13. The apparatus of claim 12, further comprising a filling mechanism
2 configured to fill stack slots that were previously spilled upon return from the
3 program method.

1 14. The apparatus of claim 13, wherein the program method is
2 associated with a single typemap to indicate a type for each variable on the
3 evaluation stack.

1 15. A computer system to facilitate code verification and garbage
2 collection in a platform-independent virtual machine, comprising:
3 a central processing unit;
4 a memory system;
5 a port for communicating with an external client;
6 a bus to couple the central processing unit, the memory system, and the
7 port;
8 a receiving mechanism within the central processing unit configured to
9 receive a code module written in a platform-independent language;
10 an examining mechanism configured to examine the code module to locate
11 a call to a program method within the code module; and

12 a transforming mechanism configured to transform the code module so
13 that all operands remaining on an evaluation stack when the program method is
14 called relate to the program method;
15 whereby verification and garbage collection of the code module is
16 simplified.

1 16. The computer system of claim 15, wherein transforming the code
2 module involves ensuring that local variables hold only values of a single type and
3 do not hold variables of different types at different times.

1 17. The computer system of claim 15, wherein transforming the code
2 module involves ensuring that the evaluation stack includes only elements related
3 to a bytecode that may trigger garbage collection when the bytecode is executed.

1 18. The computer system of claim 15, wherein transforming the code
2 module involves ensuring that only parameters for the program method are on the
3 evaluation stack when the program method is called.

1 19. The computer system of claim 15, further comprising a spilling
2 mechanism configured to spill to memory stack slots that do not include operands
3 for the call to the program method when transforming the code module.

1 20. The computer system of claim 19, further comprising a filling
2 mechanism configured to fill stack slots that were previously spilled upon return
3 from the program method.

- 1 21. The computer system of claim 20, wherein the program method is
- 2 associated with a single typemap to indicate a type for each variable on the
- 3 evaluation stack.